

Watermarking PDF Documents using Various Representations of Self-inverting Permutations

Maria Chroni and Stavros D. Nikolopoulos

Department of Computer Science & Engineering,
University of Ioannina
GR-45110 Ioannina, Greece
{mchroni,stavros}@cs.uoi.gr

Abstract. This work provides to web users copyright protection of their Portable Document Format (PDF) documents by proposing efficient and easily implementable techniques for PDF watermarking; our techniques are based on the ideas of our recently proposed watermarking techniques for software, image, and audio, expanding thus the digital objects that can be efficiently watermarked through the use of self-inverting permutations. In particular, we present various representations of a self-inverting permutation π^* namely 1D-representation, 2D-representation, and RPG-representation, and show that these representations can be efficiently applied to PDF watermarking. Indeed, we first present an audio-based technique for marking a PDF document T by exploiting the 1D-representation of a permutation π^* , and then, since pages of a PDF document T are 2D objects, we present an image-based algorithm for encoding π^* into T by first mapping the elements of π^* into a matrix A^* and then using the information stored in A^* to mark invisibly specific areas of PDF document T . Finally, we describe a graph-based watermarking algorithm for embedding a self-inverting permutation π^* into the document structure of a PDF file T by exploiting the RPG-representation of π^* and the structure of a PDF document. We have evaluated the embedding and extracting algorithms by testing them on various and different in characteristics PDF documents.

Keywords. Watermarking techniques; Text watermarking; PDF documents, Self-inverting permutations; Representations of permutations; Embedding algorithms; Extracting algorithms.

1 Introduction

Information age has altered the way people communicate by breaking the barriers imposed on communications by time, distance, and location and has undoubtedly impact not only humans activities but also global industry and economy. Communication has been greatly affected by the constant and rapid evolution of many technologies such as fiber optic, cellular and satellite technology, networking, digital transmission and compression as well as advanced computers, and

improved human-computer interaction. The aforementioned technologies allow the rapid transmission, and store, of great amounts of information.

The digital era has already had extensive impacts on business, commerce, education, services, and social life. The concepts of e-government, e-learning, e-commerce, e-business, e-publishing, refer peoples' interaction in the digital world. In this world, people everyday, interact by exchanging e-mails, instant messages, video, audio, images, and digital documents. Part of the information transmitted is an increasing amount of sensitive information, such as personal data, medical and financial records, business information, government data, legal documents. Another part of information available in the web is used to promote ones' work or product.

Electronic document, is an extensively used medium traveling over the internet for information exchange and due to the ease of copying and distributing they are susceptible to threats like illegal copying, redistribution of copyrighted documents, and plagiarism. Subsequently, it has become more important to protect the electronic documents from any malicious user while existing in the digital world. Copyright protection of digital contents is such a need of time which cannot be overlooked. In past, various methods like encryption, steganography and watermarking has been used to solve these problems. However, digital watermarking is the better solution for copyright protection than encryption and steganography. It is well known that digital watermarking methods are efficient enough to identify the original copyright owner of the contents.

Recall that there are many reasons why you would want to use watermarks in digital documents: as a copying deterrent, as a means of identifying the source of a printed document, as a means of determining whether a document has been altered, etc.

Attacks. Any action that a user can perform on a text that can affect the watermark, or its usefulness, is called attack. In [32] existing attacks on text watermarking can be classified into three main categories:

- watermark attacks,
- geometric attacks, and
- system attacks.

In a watermark attack, the adversary aims to detect and destroy the watermark, without necessarily decoding the original message. In contrast to watermark attacks, geometrical attacks are blind attacks on watermarked text documents. The process of these attacks requires neither the algorithmic knowledge of the watermarking technique nor the watermarking key, geometrical attacks intend not to remove the embedded watermark itself, but to prevent it from serving its intended purpose through altering format or content of the watermarked text documents. This type of attack includes reformatting, reproducing, sentences swapping, paragraphs shuffling, the addition/deletion of words, sentences and paragraphs. System attacks use several signal processing tools such as principal

component analysis, independent component analysis, clustering, vector quantization, etc.

Related Work. Text watermarking is the area of research that has emerged after the development of Internet and communication technologies; we mention that the first reported effort on marking documents dates back to 1993 [23].

Generally, we can classify the previous work on digital text watermarking in the following four categories:

- image based approach,
- syntactic approach,
- semantic approach, and
- structural approach.

In image-based approach, a watermark is embedded in text image. Brassil, et al. were the first to propose a few text watermarking methods utilizing text image [4,5]; they also developed document watermarking schemes based on line shifts, word shifts as well as slight modifications to the characters [6]. Maxemchuk, et al. [23,24,25] analyzed the performance of these methods, while later Low, et al. [16,17] further analyzed their efficiency. Huang and Yan [14] proposed a text watermarking method based on an average inter-word distance in each line.

In syntactic approach, the syntactic structure of the text is used to embed watermark. Atallah, et al. [3] proposed several methods of natural language watermarking, which opened up a brand-new and challenging research direction for text watermarking. Meral et al. performed morpho-syntactic alterations to the text to watermark it [26]; they also provided an overview of available syntactic tools for text watermarking [27].

In semantic approach, semantics of text are used to embed the watermark in text. Atallah et al. were the first to propose the semantic watermarking schemes [3]. Later, the synonym substitution method was proposed, in which watermark was embedded by replacing certain words with their synonyms [30]. Sun, et al. [29] proposed noun-verb based technique for text watermarking which used nouns and verbs parsed by semantic networks. Topkara, et al. proposed an algorithm of the text watermarking by using typos, acronyms and abbreviation in the text to embed the watermark [31]. Algorithms were developed to watermark the text using the linguistic approach of presuppositions [28] in which the discourse structure, meaning, and representations are observed and utilized to embed watermark bits. The text pruning and the grafting algorithms were also developed in the past. Another algorithm based on text meaning representation (TMR) strings has also been proposed [18].

The structural approach is the most recent approach used for copyright protection of text documents. In this approach, text is not altered, rather it is used to logically embed watermark in it. A text watermarking algorithm, for copyright protection of text using occurrences of double letters (aa-zz) in text, has recently been proposed [15]. Recently, a significant number of techniques have

been proposed in the literature which use Portable Document Format (PDF) files as cover media in order to hide data [7,8,19,20,21,22,33].

Contribution. In this paper, in order to provide to web users copyright protection of their digital documents, we present easily implemented techniques for watermarking PDF documents. Our aim is to extent the digital objects that the proposed representations of a self-inverting permutation, i.e. the 1D-representation, the 2D-representation, and the RPG-representation, can be efficiently applied to; note that, RPG-representation means the encoding of permutation π^* as a reducible permutation graph $F[\pi^*]$.

We first propose an image-based technique for marking the PDF document T by exploiting the 1D-representation of a permutation π^* . The embedding of a mark is performed by increasing the distance (or, space) between two consecutive words in a paragraph of the document T . The extraction algorithm operates in a reverse manner.

Consequently, since pages of a PDF documents T are two dimensional objects, we propose an algorithm for encoding a self-inverting permutation π^* into a document T by first mapping the elements of π^* into an $n^* \times n^*$ matrix A^* and then using the information stored in A^* to mark invisibly specific areas of PDF document T resulting thus the watermarked PDF document T_w . We also propose an efficient algorithm for extracting the embedded self-inverting permutation π^* from the watermarked PDF document T_w by locating the positions of the marks in T_w ; it enables us to recontract the 2D representation of the self-inverting permutation π^* .

Finally, we describe a watermarking algorithm for embedding a self-inverting permutation into the document structure of a PDF file T , by exploiting the graph representation of π^* and the structure of a PDF document T . More precisely, in light of the two embedding algorithms `Encode_SiP.to.RPG-I` and `-II`, we present an algorithm for embedding a reducible permutation graph $F[\pi^*]$ into a PDF document T . The main idea behind the proposed embedding algorithm is a systematic addition of appropriate object-references in the input PDF document T , through the use of entries of type `\kye()`, so that the graph $F[\pi^*]$ can be easily constructed from the page tree $PT(T_w)$ of the resulting watermarked document T_w .

Road Map. The paper is organized as follows: In Section 2 we establish the notation and related terminology, and we present background results. In Section 3 based on the three different representations of self-inverting permutation (SiP), i.e., the 1D-representation, the 2D-representation, and the RPG-representation (the encoding of permutation π^* as a reducible permutation graph $F[\pi^*]$), we present the algorithms `Embed_SiP.to.PDF-I`, `Embed_SiP.to.PDF-II`, and `Embed_RPG.to.PDF`, along with the corresponding extracting algorithms, for embedding a watermark number (or, equivalently, a self-inverting permutation π^* or a reducible permutation graph $F[\pi^*]$) into a PDF document file. Finally, in Section 4 we conclude the paper and discuss possible future extensions.

2 Background Results

In this section we give some definitions and the theoretical background we use towards the watermarking of Portable Document Format (PDF) documents. We first briefly present the different representations of a self-inverting permutation (SiP), and then we present the structure of PDF documents.

1D-representation of SiP. Recently, we presented the one-dimensional representation (1D-representation) of a self-inverting permutation (SiP) π^* and the one-dimensional marked representation of π^* (1DM-representation), and showed how to embed a SiP, represented by 1D space, into an audio signal [9,10]. In our 1D-representation, the elements of the permutation π are mapped in specific cells of an array B of size n^2 as follows:

- number $\pi_i \longrightarrow$ entry $B((\pi_i^{-1} - 1)n + \pi_i)$

or, equivalently, the cell at the position $(i - 1)n + \pi_i$ is labeled by the number π_i , for each $i = 1, 2, \dots, n$.

In our 1DM representation, a permutation π over the set N_n is represented by an n^2 array B^* by marking the cell at the position $(i - 1)n + \pi_i$ by a specific symbol, where, in our implementation, the used symbol is again the asterisk character “*”.

2D-representation of SiP. We have also presented the two-dimensional representation of a SiP (2D-representation) and the two-dimensional marked representation of SiP (2DM-representation); note that, these representations have been recently used for watermarking images in the frequency domain [9,10].

We defined the 2D-representation of a SiP as the representation where the elements of the permutation $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ are mapped in specific cells of an $n \times n$ matrix A as follows:

- number $\pi_i \longrightarrow$ entry $A(\pi_i^{-1}, \pi_i)$

or, equivalently,

- the cell at row i and column π_i is labeled by the number π_i , for each $i = 1, 2, \dots, n$.

In 2DM-representation the cell at row i and column π_i of matrix A is marked by a specific symbol, for each $i = 1, 2, \dots, n$.

We have presented algorithms for embedding the 2D-dimensional representation of SiP in an image. Recall that the matrix A incorporates important structural properties which, in image watermarking, make it possible to detect geometric transformations on the watermarked image. The properties of the matrix A are the following:

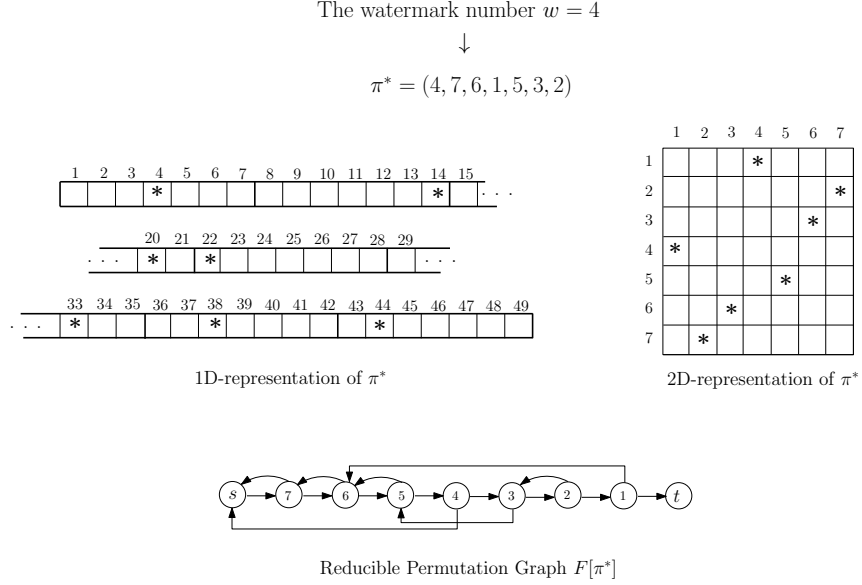


Fig. 1. Three different representations of permutation $\pi^* = (4, 7, 6, 1, 5, 3, 2)$.

- the matrix A is symmetric;
- the main diagonal of the symmetric matrix A^* has always one and only one marked cell;
- the marked cell on the diagonal is always in entry (i, i) of A^* , where $i = \lceil \frac{n^*}{2} \rceil + 1, \lceil \frac{n^*}{2} \rceil + 2, \dots, n^*$.

The authors of this paper, we have also presented an efficient and easily implemented algorithm for encoding numbers as reducible permutation graphs (or, for short, RPG) through the use of self-inverting permutations [12,13]. In particular, we have proposed two such encoding algorithms: the algorithm **Encode_SiP.to.RPG-I** applies to any permutation π and relies on domination relations on the elements of π whereas the algorithm **Encode_SiP.to.RPG-II** applies to a self-inverting permutation π^* produced in any way and relies on the decreasing subsequences of π^* . Figure 1 summarizes by an example the representations of the permutation $\pi^* = (4, 7, 6, 1, 5, 3, 2)$.

2.1 Structure of a PDF Document

The Portable Document Format (PDF) [2] is an open standard (defined in ISO 32000) which facilitates device and platform independent capture and representation of rich information such as text, multimedia and graphics, into a single

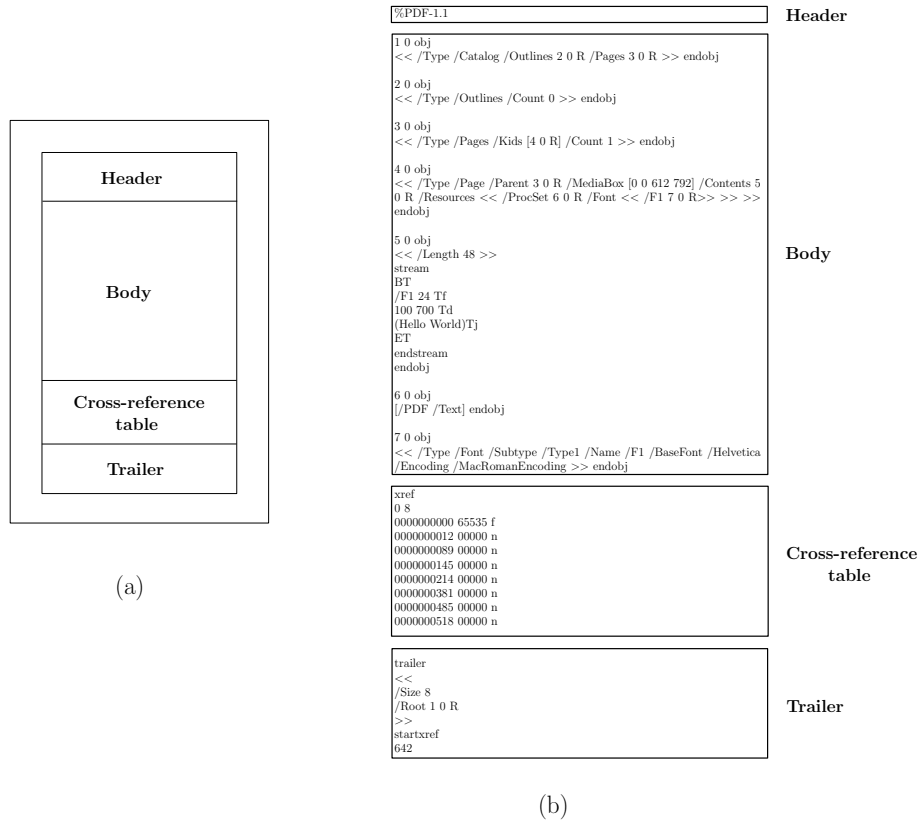


Fig. 2. (a) The structure of a PDF file; (b) The code of a PDF file containing, in object 5 0 obj, the text “Hello World”.

medium. Thus the PDF format enables viewing and printing of a rich document, independent of either application software or hardware. In this section we present a structural analysis of a PDF file, by giving its basic components.

Object. An object is the basic element in PDF files, in which eight kinds of objects, namely Boolean Object, Numeric Object, String Object, Name Object, Array Object, Null Object, Dictionary and Stream Object are sustained. Objects may be labeled so that they can be referred to by other objects. A labeled object is called an indirect object.

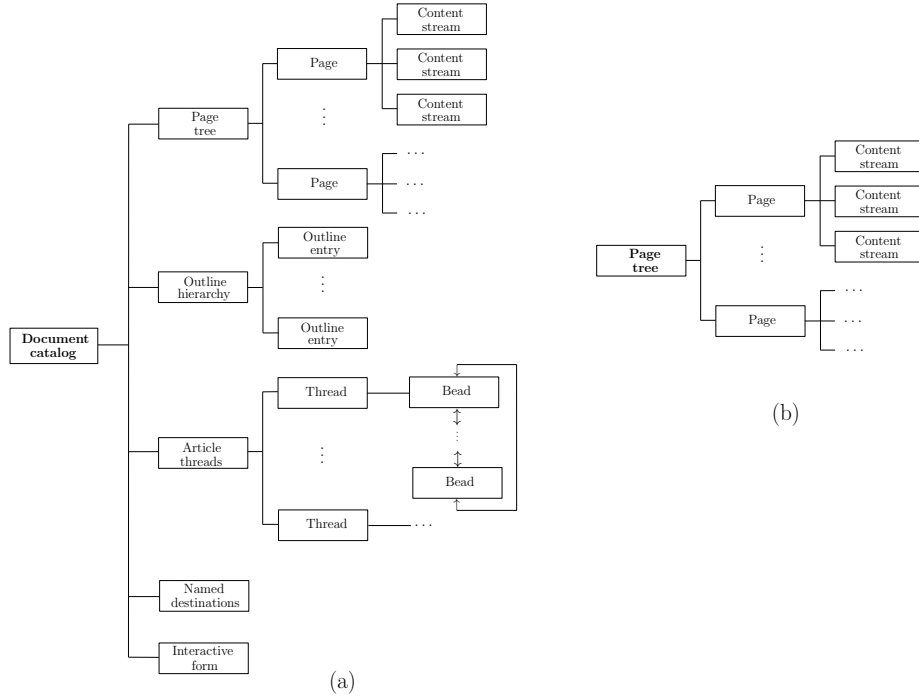


Fig. 3. (a) The main structural components of a PDF file; (b) The document structure of PDF file.

File structure. The PDF file structure determines how objects are stored in a PDF file, how they are accessed, and how they are updated. The file structure (see, Figure 2) includes the following:

- an one-line header identifying the version of the PDF specification to which the file conforms,
- a body containing the objects that make up the document contained in the file,
- a cross-reference table containing information about the indirect objects in the file, and
- a trailer giving the location of the cross-reference table and of certain special objects within the body of the file.

Figure 2 shows an example of a PDF file and its internal file structure.

Document structure. The PDF document structure specifies how the basic object types are used to represent components of a PDF document: pages, fonts, annotations, and so forth. The document structure of a PDF file is organized in the shape of an object tree topped by Catalog, Page tree, Outline hierarchy and Article thread included. The Outline hierarchy is the bookmark of PDF, and Page tree includes page and Pages which in turn includes the total page number and each page marker. Page, the main body of PDF file, is the most important object which involves the typeface applied, the text, pictures, page size, and so on. The organization of other objects is analogous to Page tree. Figure 3 illustrates the structure of the object hierarchy.

3 Watermarking PDF Documents

In this section we describe embedding algorithms for encoding a SiP π^* into a digital document T . More specifically, we embed the permutation π^* into a PDF document by exploiting (i) the one-dimensional representation of π^* , (ii) the two-dimensional representation of a π^* , and (iii) the encoding of π^* as a reducible permutation graph $F^*[\pi^*]$.

3.1 Embed Watermark into PDF - I

We first design an embedding algorithm for watermarking a PDF document by exploiting the 1D-representation of a permutation π^* . The marking is performed by increasing the space between two consecutive words in a paragraph of T .

Let B^* be the 1D array of size $n = n^* \times n^*$ which represents the permutation π^* of length n^* , and let $(w_1, s_1), (w_2, s_2), \dots, (w_n, s_n)$ be n pairs of type “word-space” of a paragraph par of the input PDF document; recall that the entry $B^*((i-1)n^* + \pi_i^*)$ contains the symbol “*”, $1 \leq i \leq n^*$. The algorithm increases by a small value “ c ” the i -th space of the pair (w_i, s_i) if $B^*((i-1)n^* + \pi_i^*) = “*”$.

We next give a high-level description, with respect to PDF modification, of our proposed embedding algorithm.

Algorithm Embed_SiP.to.PDF-I

1. Compute the 1DM representation of the permutation π^* , i.e., construct the array B^* of size $n = n^* \times n^*$ where the $(i-1)n^* + \pi_i^*$ entry of B^* contains the symbol “*”, $1 \leq i \leq n^*$;
2. Select an appropriate paragraph par on a page P of PDF document T to embed the self-inverting permutation π^* ;
3. Partition the paragraph par into n pairs $(w_1, s_1), (w_2, s_2), \dots, (w_n, s_n)$, where w_i and s_i are the i -th word and space, respectively, in selected paragraph par , $1 \leq i \leq n$;
4. For each pair (w_i, s_i) s.t. $B^*((i-1)n^* + \pi_i^*) = “*”$, increases the space s_i or, equivalently, distance $d(w_i, w_{i+1})$ between words w_i and w_{i+1} , by a relative small value c , $1 \leq i \leq n$;

5. Return the watermarked PDF document T_w .

- $d(w_i, w_{i+1}) > d(w_{i-1}, w_i)$, or
- $d(w_i, w_{i+1}) > d(w_{i+1}, w_{i+2})$

3.2 Embed Watermark into PDF - II

The main idea behind the embedding algorithm, we call it **Embed_SiP.to.PDF-II**, is similar of that of algorithm **Embed_SiP.to.Image-F** (see, [11]). The most important of this idea is the fact that it suggests a way in which the permutation π^* can be represented with a 2D-representation and since pages of a PDF document T are two dimensional objects that representation can be efficiently marked on them resulting the watermarked PDF document T_w ; in a similar way as in our image watermarking approach, such a 2D-representation can be efficiently extracted for a watermarked PDF document T_w and converted back to the self-inverting permutation π^* .

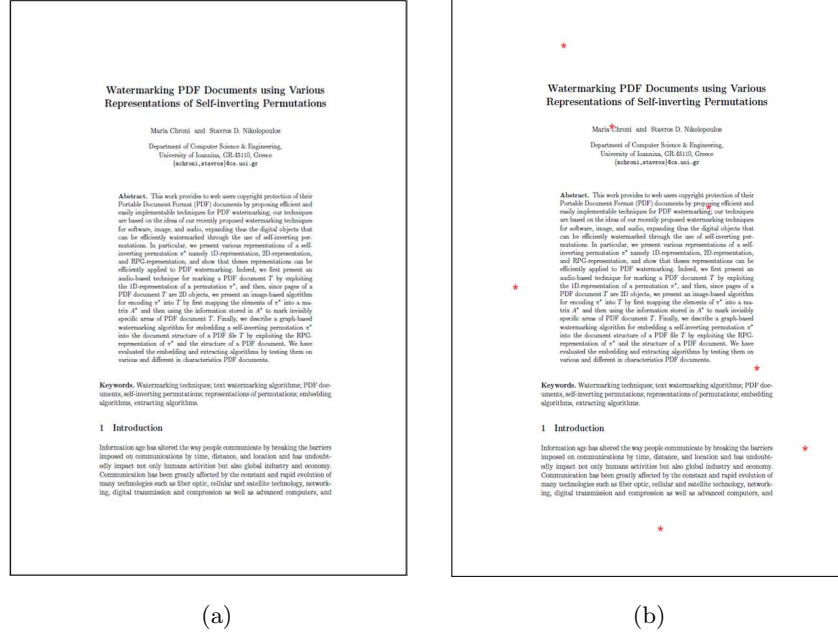


Fig. 5. (a) The initial PDF document T ; (b) The watermarked PDF document T_w using the 2D representation of permutation $\pi^* = (4, 7, 6, 1, 5, 3, 2)$; the red stars indicate the marks.

Let A^* be the 2D matrix of size $n^* \times n^*$ which represents the permutation π^* of length n^* . The marking of the input PDF document T is performed by selecting an appropriate page P of T and setting n^* objects (e.g., characters, symbols, images) in a specific positions on page P , $1 \leq i \leq n^*$. In fact, we set an object \mathcal{O}_i in position with (x'_i, y'_i) coordinates on page P if $A^*(x_i, y_i) = *$, where $1 \leq x_i, y_i \leq n^*$ and $0 \leq x'_i, y'_i \leq \text{size}(P)$; note that, $(0, 0)$ is the lower-left point (or, equivalently, the bottom-left corner) of the page P .

The algorithm takes as input a SiP π^* and a PDF document T , and returns the watermarked document T_w ; it consists of the following steps.

Algorithm Embed_SiP.to.PDF-II

1. Compute the 2DM representation of the self-inverting permutation π^* , i.e., construct an array A^* of size $n^* \times n^*$ s.t. the entry $A^*(i, \pi_i^*)$ contains the symbol “*”, $1 \leq i \leq n^*$;
2. Select an appropriate page P to embed the permutation π^* and compute the size $\text{size}(P)$ of the page P , say, $N \times M$;

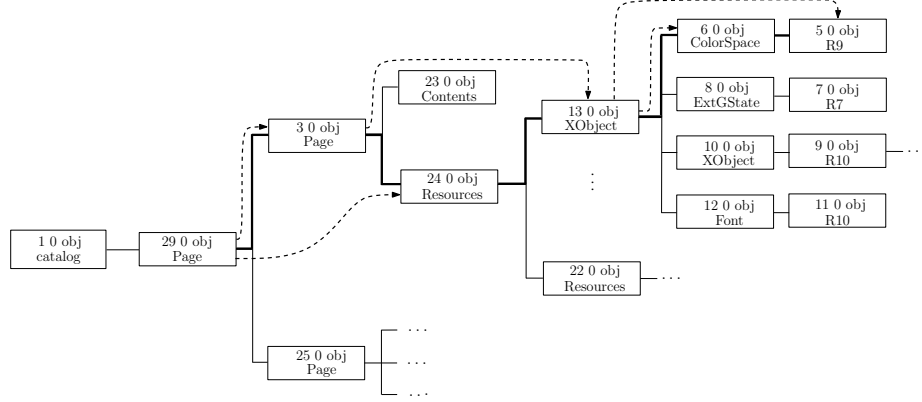


Fig. 6. The watermarked $DS(T_w)$ which encodes the RPG of $\pi^* = (4, 5, 3, 1, 2)$.

3. Segment the PDF page P into $n^* \times n^*$ grid-cells C_{ij} of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$, $1 \leq i, j \leq n^*$;
4. For each grid-cell C_{ij} s.t. $A^*(i, j) = *$, mark the cell C_{ij} by setting a symbol, with an appropriate color, in any position inside C_{ij} of P , $1 \leq i, j \leq n^*$, resulting thus the marked document T_w ;
5. Return the watermarked PDF document T_w .

Extraction. The algorithm which extracts the permutation π^* from the watermarked PDF T_w operates in a similar way as the corresponding extraction algorithm for images: it takes the input watermarked image I_w , locate the marked page P , computes its $N \times M$ size, and segments P into $n^* \times n^*$ grid-cells C_{ij} of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$; then, it computes the permutation π^* by finding the coordinates (x_i, y_i) of the n^* symbols in the page P , $1 \leq i \leq n^*$; we call it `Extract_PDF.from_SiP-II`.

3.3 Embed an RPG into a PDF

In this section we describe a watermarking algorithm for embedding a self-inverting permutation π^* into a PDF document T , by exploiting the RPG-representation of π^* and the structure of a PDF document T .

Indeed, we have recently proposed two algorithms, namely `Encode_SiP.to.RPG-I` and `-II`, for encoding self-inverting permutations π^* as reducible permutation graphs $F[\pi^*]$. Moreover, in this paper we have described the document structure $DS(T)$ of a PDF document T (see, Subsection 2.1); note that, the document

structure of a PDF file always contains a node, namely **Document-catalog**, and a page tree $PT(T)$ rooted at node **Page-tree**, denoted by $root(pt)$; see, Figure 3(b).

In light of the two encoding algorithms **Encode_SiP.to.RPG-I** and **-II**, we next present an algorithm for embedding a reducible permutation graph $F[\pi^*]$ into a PDF document T . The main idea behind the proposed embedding algorithm is a systematic addition of appropriate object-references in selected nodes of the page-tree $PT(T)$ of the document structure $DS(T)$, through the use of entries of type `/Kye(·)`, so that the graph $F[\pi^*]$ can be easily constructed from the page-tree $PT(T_w)$ of the resulting watermarked document T_w .

Let $F[\pi^*]$ be a reducible permutation graph produced by one of our two encoding algorithms (i.e., **Encode_SiP.to.RPG-I** or **-II**), and let $u_{n+1}, u_n, \dots, u_1, u_0$ be the nodes of the graph $F[\pi^*]$; note that, $F[\pi^*]$ does not contain the back-edge (u_0, u_{n+1}) . In order to simplify the extraction process, the graph $F[\pi^*]$ which is embedded into a PDF document T contains one extra back-edge, i.e., the edge (u_0, u_{n+1}) ; see, [12,13].

The algorithm for embedding a reducible permutation graph $F[\pi^*]$ into a PDF document T is called **Encode_RPG.to.PDF** and is described below.

Algorithm Encode_RPG.to.PDF

1. Compute the document structure $DS(T)$ of the input PDF document T and locate its page-tree $PT(T)$; let **node(dc)** be the document catalog node of structure $DS(T)$ and $root(pt)$ be the root node of the page tree $PT(T)$; see, Figure 3(b);
2. Compute a path $O(T) = (v_{n+1}, v_n, \dots, v_1, v_0)$ on $n + 2$ nodes (i.e., objects) of the page-tree $PT(T)$ s.t. $v_{n+1} = root(pt)$, and set $s = v_{n+1}$ and $t = v_0$;
3. Assign an exact pairing (i.e., 1-1 correspondence) of the $n + 2$ nodes of path $O(T)$ to the nodes $u_{n+1}, u_n, \dots, u_1, u_0$ of the watermark graph $F[\pi^*]$;
4. For each back-edge (u_i, u_j) of the graph $F[\pi^*]$ (i.e., $u_j > u_i$), add the forward-edge (v_j, v_i) in page-tree $PT(T)$ by adding in object $[v_j \ 0 \ obj]$ an entry of type `/Key(vi 0 R)`; add in object $[v_{n+1} \ 0 \ obj]$ an entry of type `/Key(v0 0 R)`;
5. Return the modified PDF document T , i.e., the watermarked document T_w ;

Let us briefly discuss the way we add forward-edge in the page-tree $PT(T)$; recall that, in Step 4 of the previous algorithm **Encode_RPG.to.PDF** we add the forward-edge (v_j, v_i) in page-tree $PT(T)$ by adding in object $[v_j \ 0 \ obj]$ an entry of type `/Key(vi 0 R)`. The entry `/Key(vi 0 R)` may be of various types; note that, `/Key(·)` is used as parameter in our algorithm's description.

In our implementation, for the forward-edge (v_j, v_i) such that the object $[v_j \ 0 \ obj]$ is not the root-node $root(pt)$ of the page-tree $PT(T)$, we always chose the entry `/Key(vi 0 R)` which we add in object $[v_j \ 0 \ obj]$ to be of the same type of object $[v_i \ 0 \ obj]$. In the case where $v_j = root(pt)$, we chose the entry `/Key(vi 0 R)` to be of type `/Kids(·)`.

For example, in Figure 6 we have added forward-edges from object [29 0 obj] to object [3 0 obj], from object [29 0 obj] to object [24 0 obj], from object [3 0 obj] to object [13 0 obj], etc. Thus, in our implementation we have added in the root-node object [29 0 obj] the entries `/Kids(3 0 R)` and `/Kids(24 0 R)`, in object [3 0 obj] the entry `/XObject(13 0 R)`, while in object [13 0 obj] the entries `/ColorSpace(6 0 R)` and `/R9(5 0 R)`.

Remark 3.1. Let T be a PDF file and let $PT(T)$ be a page-tree of the document structure $DS(T)$. A node of the page-tree $PT(T)$ may contain several entries `/Key(·)` of various types. We mention that, some types are required for the entries in specific nodes of $PT(T)$; for example, the required entries in the root-node $root(pt)$ of the page-tree $PT(T)$ are the following four: `/Type(·)`, `/Parent(·)`, `/Kids(·)`, and `/Count(·)`.

Extraction. We next describe the corresponding extraction algorithm, which we call `Extract_RPG.from.PDF`; it extracts the graph $F[\pi^*]$ from the PDF document T_w watermarked by the embedding algorithm `Encode_RPG.to.PDF`. The algorithm works as follows:

- Take first as input the PDF document T_w watermarked by the embedding algorithm `Encode_RPG.to.PDF`, compute the document structure $DS(T_w)$ of T_w , and locate its page tree $PT(T_w)$; then, find in object $root(pt)$, where $root(pt)$ is the root of the tree $PT(T_w)$, the entry `/Kids(v_k 0 R)` s.t. v_k is not a child of $root(pt)$, and set $v_{n+1} = root(pt)$ and $v_0 = v_k$;
- Compute the path $O(T) = (v_{n+1}, v_n, \dots, v_1, v_0)$ of $PT(T_w)$, from node $root(pt)$ to v_0 , and assign an exact pairing (i.e., 1-1 correspondence) of the $n+2$ nodes of path $O(T)$ to the nodes $u_{n+1}, u_n, \dots, u_1, u_0$ of a graph $F[\pi^*]$; initially, $E(F[\pi^*]) = \emptyset$;
- Add edges (u_{i+1}, u_i) in $F[\pi^*]$ for $i = n, n-1, \dots, 0$, and the edge (u_i, u_j) iff (v_i, v_j) is a forward edge in the page tree $PT(T_w)$;
- Delete the edge (u_{n+1}, u_0) from the graph $F[\pi^*]$;
- Return the graph $F[\pi^*]$;

It is easy to see that, by construction the returned graph $F[\pi^*]$ is a reducible permutation graph produced by either algorithm `Encode_SiP.to.RPG-I` or algorithm `Encode_SiP.to.RPG-II`. Thus, $F[\pi^*]$ has the following property: the structure which results after deleting

- (i) all the forward edges (u_{i+1}, u_i) of $F[\pi^*]$, $0 \leq i \leq n$, and
- (ii) the node u_0

is either the tree $T_d[\pi^*]$ or the tree $T_s[\pi^*]$ produced during the execution of either the decoding algorithm `Decode_RPG.to.SiP-I` or algorithm `Decode_RPG.to.SiP-II`, respectively (see, [9,12,13]). Thus, we can efficiently extract the self-inverting permutation π^* embedded into a PDF document T by algorithm `Encode_RPG.to.PDF`.

4 Concluding Remarks

In this paper we presented embedded algorithms, along with their corresponding extraction algorithms, for watermarking PDF documents T using three different representations of a self-inverting permutation π^* , namely 1D-representation, 2D-representation, and RPG-representation; note that, RPG-representation means the encoding of permutation π^* as a reducible permutation graph $F[\pi^*]$.

The main features of our algorithms, i.e., the way they mark a PDF document T or, equivalently, the way they embed a self-inverting permutation π^* into document T , are summarized as follows:

- In the first algorithm **Embed_SiP.to.PDF-I** the marking of a PDF document T is performed by increasing the distance (or, space) between two consecutive words in a paragraph of T .
- The main idea behind the second algorithm **Embed_SiP.to.PDF-II** is based on the fact that π^* has a 2D-representation and, since pages of a PDF documents T are two dimensional objects, it can be efficiently used to mark specific positions on a page of T resulting thus the watermarked PDF document T_w .
- The third graph-based embedding algorithm **Encode_RPG.to.PDF** uses a different approach: it exploits the structure of a PDF document T and embeds the graph $F[\pi^*]$ into T by adding appropriate object-references in the document T , through the use of entries of type `/Kids(k 0 R)`, so that the graph $F[\pi^*]$ can be easily constructed from the page tree $PT(T_w)$ of the resulting watermarked document T_w .

In light of our graph-based embedding algorithm **Encode_RPG.to.PDF** it would be very interesting to investigate the possibility of altering other components of the document structure of a PDF file in order to embed the graph $F[\pi^*]$; we leave it as a direction for future work.

Moreover, an interesting open question is whether the embedding approaches and techniques used in this paper can help develop efficient encoding algorithms having “better” properties with respect text attacks; we leave it as an open problem for future investigation.

References

1. Amano, T., and Misaki, D.: A feature calibration method for watermarking of document images. In: IEEE Proceedings of the 5th In'l Conference on Document Analysis and Recognition (ICDAR'99), pp. 91–94 (1999)
2. Adobe Systems Incorporated. Adobe Portable document format Version 1.7, <http://www.adobe.com>, Nov. 2006
3. Atallah, M.J., Raskin, V., Hempelmann, C.F., Karahan, M., Sion, R., Topkara, U., and Triezenberg, K.E.: Natural language watermarking and tamperproofing. LNCS 5, Springer, 196–212 (2003)

4. Brassil, J.T., Low, S., Maxemchuk, N.F., and Gorman, L.O.: Electronic Marking and Identification Techniques to Discourage Document Copying. *IEEE Journal on Selected Areas in Communications* 13(8), 1495–1504 (1995)
5. Brassil, J.T., Low, S., Maxemchuk, N.F., Gorman, L.O.: Hiding information in document images. In: *Proceedings of the 29th Annual Conference on Information Sciences and Systems*, Johns Hopkins University, pp. 482–489 (1995)
6. Brassil, J.T., Low, S., and Maxemchuk, N.F.: Copyright protection for the electronic distribution of text documents. In: *IEEE Proceedings* 87(7), pp. 1181–1196 (1999)
7. Bindra, G.S.: Invisible communication through Portable Document File (PDF) format. In: *Proceedings of the 7th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'11)*, pp. 173–176 (2011)
8. Bindra, G.S.: Masquerading as a trustworthy entity through Portable Document File (PDF) format. In: *Proceedings of the 2011 IEEE International Conference on PASSAT and IEEE International Conference on Social Com.*, Boston, USA, pp. 784–789 (2011)
9. Chroni, M.: Algorithmic Techniques for Encoding Permutations and Permutation Graphs for Watermarking Software, Image, Audio, and Text. Department of Computer Science and Engineering, University of Ioannina, PhD Thesis (2014).
10. Chroni, M., Fylakis, A., and Nikolopoulos, S.D.: From image to audio watermarking using self-inverting permutations. In: *Proceedings of the 10th Int'l Conference on Web Information Systems and Technologies (WEBIST'14)*, SciTePress, pp. 177–184 (2014)
11. Chroni, M., Fylakis, A., and Nikolopoulos, S.D.: Watermarking images in the frequency domain by exploiting self-inverting permutations. In: *Proceedings of the 9th Int'l Conference on Web Information Systems and Technologies (WEBIST'13)*, SciTePress, pp. 45–54 (2013)
12. Chroni, M., and Nikolopoulos, S.D.: Design and evaluation of a graph codec system for software watermarking. In: *Proceedings of the 2nd Int'l Conference on Data Management Technologies and Applications (DATA'13)*, SciTePress, 277–284 (2013)
13. Chroni, M., and Nikolopoulos, S.D.: An efficient graph codec system for software watermarking. In: *Proceedings of the 36th Int'l Conference on Computers, Software, and Applications (COMPSAC'12)*, Workshop STPSA'12, IEEE, pp. 595–600 (2012)
14. Huang, D., and Yan, H.: Interword distance changes represented by sine waves for watermarking text images. *IEEE Trans. Circuits and Systems for Video Technology* 11(12), 1237–1245 (2001)
15. Jalil, Z., and Mirza, A.M.: An invisible text watermarking algorithm using image watermark. In: *Proceedings of the Innovations in Computing Sciences and Software Engineering*, pp. 147–152 (2010)
16. Low, S.H., Maxemchuk, N.F., and Lapone, A.M.: Document identification for copyright protection using centroid detection. *IEEE Transactions on Communications* 46(3), 372–381 (1998)
17. Low, S.H., and Maxemchuk, N.F.: Capacity of text marking channel. *IEEE Signal Processing Letters* 7(12), 345–347 (2000)
18. Lu, P., Lu, Z., Zhou, Z., and Gu, J.: An optimized natural language watermarking algorithm based on TMR. In: *Proceedings of the 9th International Conference for Young Computer Scientists*, pp. 1459–1463 (2008)

19. Liu, H., Li, L., Li, J., and Huang, J.: Three novel algorithms for hiding data in pdf files based on incremental updates. *Digital Forensics and Watermarking*, Springer Berlin Heidelberg, pp. 167–180 (2012)
20. Liu, X., Zhang, Q., Tang, C., Zhao, J., and Liu, J.: A Steganographic Algorithm for Hiding Data in PDF Files Based on Equivalent Transformation. In: *International Symposiums on Information Processing (ISIP'08)*, pp. 417–421 (2008)
21. Liu, Y., Sun, X., and Luo, G.: A novel information hiding algorithm based on structure of PDF document. *Computer Engineering* 32(17), 230–232 (2006)
22. Lee, I.S., and Tsai, W.H.: A new approach to covert communication via PDF files. *Signal Processing* 90(2), 557–565 (2010)
23. Maxemchuk, N.F., and Low, S.: Marking text documents. In: *Proceedings of the IEEE International Conference on Image Processing*, pp. 13–16 (1997)
24. Maxemchuk, N.F., and Low, S.H.: Performance comparison of two text marking methods. *IEEE Journal of Selected Areas in Communications* 16(4), 561–572 (1998)
25. Maxemchuk, N.F.: Electronic document distribution. *AT&T Technical Journal* 73(5), 73–80 (1994)
26. Meral, H.M., and Sankur, B., Özsoy, A., Güngör, T., and Sevinç, E.: Natural language watermarking via morphosyntactic alterations. *Computer Speech and Language* 23(1), 107–125 (2009)
27. Meral, H.M., Sevinç, E., Ünkar, E., Sankur, B., Özsoy, A., and Güngör, T.: Syntactic tools for text watermarking. In: *Proceedings of the 19th SPIE Electronic Imaging Conference on Security, Steganography, and Watermarking of Multimedia Contents*, San Jose, CA, (2007)
28. Macq, B., and Vybornova, O.: A method of text watermarking using presuppositions. *Electronic Imaging 2007*, Society for Optics and Photonics, 65051R–65051R–10 (2007)
29. Sun, X., and Asiimwe, A.J.: Noun-verb based technique of text watermarking using recursive decent semantic net parsers. *LNCS* 3612, 958–961 (2005)
30. Topkara, U., Topkara, M., and Atallah, M.J.: The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions. In: *Proceedings of ACM Multimedia and Security Conference* (2006)
31. Topkara, M., Topraka, U., and Atallah, M.J.: Information hiding through errors: A confusing approach. In: *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, San Jose, CA, (2007)
32. Zhou, X., and Zhao, W., and Wang, Z., and Pan, L.: Security theory and attack analysis for text watermarking. In: *Proceedings of the International Conference on E-Business and Information System Security (EBISS)*, pp. 1–6 (2009)
33. Zhong, S., Cheng, X., and Chen, T.: Data hiding in a kind of PDF texts for secret communication. *International Journal of Network Security* 4(1), 17–26 (2007)